1994

Algorithm for cartesian trajectory planning of a PUMA 560 robot

Dwaraka Srinivasan
Lehigh University

AUTHOR:

Srinivasan, Dwaraka

TITLE:

Algorithm for Cartesian Trajectory Planning for a Puma 560 Rebot

DATE: May 29, 1994

Lehigh University

Algorithm for Cartesian Trajectory Planning for a PUMA 560 Robot

THESIS

SUBMITTED TO THE GRADUATE SCHOOL IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science
in
Industrial Engineering

by

Dwaraka Srinivasan

Bethlehem
Pennsylvania 18015
May 1994

Certificate of Approval

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

5/12/94
Date

Professor in Charge

Chairperson of Deparment

Acknowledgements

I thank the department of Industrial Engineering for recommending me for a tuition fee scholarship during my graduate studies at Lehigh University. I would also like to express my gratitude to the Graduate School and the College of Engineering and Applied Sciences for granting me a scholarship and waiving my tuition during my studies at Lehigh.

I would like to thank Dr. Gregory Tonkay for his guidance and patience during this study.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Trajectory planning is concerned with the development of time schedules for position, velocity and acceleration of either the end effector or the joints of a robot. Trajectory planning in cartesian space involves highly complex computations and this has hindered the implementation of cartesian space trajectory planning. On the other hand, trajectory planning in joint space is relatively simpler and makes control of the robot easier since control is done at the joint level. Interpolation of the joint angles does not necessarily imply that the trajectory followed by the end effector of a robot is the same as the cartesian trajectory originally specified. The objective of this thesis is the development and simulation of a joint space path planning algorithm that approximates the cartesian path within specified tolerances. In order to ensure that the joint space trajectory is being followed, the joint space path must be tracked. A simple algorithm has been developed for this also.

Based on the curvature of the cartesian trajectory, a systematic procedure is developed for selecting knot points on the cartesian path and subsequent approximation of the cartesian path by concatenated line segments. The knot points are then transformed to joint space and splined, taking into account the continuity constraints. The inputs required for the study are a complete cartesian space trajectory parameterized with respect to time and the beginning and ending velocity and acceleration. Also, a simple method is proposed for continuous tracking of the path in joint space. The algorithm is implemented by simulating a PUMA 560 robot. These proposed procedures are simple and spline the knot points satisfactorily.

1 Introduction

1.1 Statement of the problem

The movements of a robot can be similar to that of the human body. Ideally we would want the robot to have the dexterity of the human hand and to follow a desired trajectory just as a human hand would do. But in reality that is not the case. The hand of a robot, also known as the end effector, is to move from one point to another in cartesian space along a desired trajectory. But due to the configuration of the robot, certain points may not be attainable. These points, known as singular points, are points at which the velocity of the joints of the robot becomes infinity.

There are two ways of obtaining a desired cartesian trajectory. One method, known as cartesian trajectory planning involves computation of all the joint angles for as many points as possible on the cartesian path. Next, these angles are transformed back to cartesian space, ensuring that the desired trajectory is obtained. At the same time singular points must be avoided. This method is computationally complex. Another method is to select certain non singular points on the cartesian trajectory, known as knot points, and transform these knot points into joint angles. The coordinates so obtained in joint space are then splined and tracked in joint space itself. The advantage of this method is that, since control of the robot is done at the joint level, this method is less computationally complex. The disadvantage is that the trajectory obtained finally in cartesian space is not necessarily the same as the trajectory initially specified.

So, the problem is as follows. Intermediate points must be selected on the cartesian space path and then transformed to joint space avoiding singularities. Then these points must be splined in joint space. The trajectory obtained in joint space, when transformed back to cartesian space, should give a close approximation of the original cartesian trajectory specified. When splining the points in joint space, the velocity and acceleration requirements must be satisfied. The joint space path must be tracked by the controller in order to ensure that there is no deviation from the path.

1.1.1 **The PUMA 560**

In this research, the proposed algorithms were implemented by simulating a PUMA 560 robot. The PUMA 560 is a rotary joint robot with six degrees of freedom. It resembles the configuration of the human arm to a large extent. The joints are numbered 1 through 6. As with many industrial robots, the joint axes of joints 4, 5 and 6 intersect at a common point. In the case of the PUMA 560, a gearing arrangement in the wrist of the robot manipulator couples together the motions of joints 4, 5 and 6. This implies that a distinction must be made for these three joints between joint space and cartesian space and the kinematics have to be solved in two steps. A detailed description of the PUMA 560 and its kinematics is given in chapter 2.

1.2 **Literature Review**

To execute trajectory planning algorithms in cartesian space, the transformation from cartesian coordinates to joint space coordinates in real time is required since control is done at the joint level. This involves complex computations. Since there is no functional transformation between a cartesian trajectory and a joint space trajectory, curve fitting techniques have been used. Paul[9] proposed that the paths be made up of straight line segments connected together by smooth transitions with controlled acceleration.The endpoints of the line segments were the intermediate knot points and in [10] these points are interpolated by joint trajectories. Taylor[10] proposed the precomputation of enough intermediate knot points in order to drive the manipulator by interpolation of joint parameter values while keeping the tool in an approximate straight line path. The errors in translation and rotation between a cartesian path and a joint space path can be easily specified to be within certain tolerances. In [4] a sequence of knot points is specified and splined using cubic splines. The total time travelled is also minimized. Cubic and quartic splines are used for interpolating knot points in [7] and approximation errors are reduced using the method of least squares.

The algorithms developed in [10], [7], [8] must be executed off line. In [2] intermediate knot points are selected based on curvature of the cartesian path. The path is approximated by concatenated straight line segments and the knot points are splined in joint space using simple and modified quartic spline interpolation. This thesis is a modification of the work done by Chang et al in [2]. The tranformations

from cartesian to joint space and vice versa are done utilizing the equations in [3]. These equations are based on Featherstone's method [4].

1.2.1 **Assumptions**

The following are the assumptions made in order to perform this research:

1. The cartesian path can be parameterized by the time variable 't' and is at least second order differentiable in t.
2. The robot used is a PUMA 560.

This assumption was necessary because the equations used for the transformation of the knot points from cartesian space to joint space and vice versa were specially developed for the PUMA 560 [3], [4]. Also the PUMA 560 is one of the most popular robots used in the industry.

3. The PUMA 560 is equipped with a stepper motor.

Though the PUMA 560 is not equipped with a stepper motor, the algorithm developed for tracking the path in joint space is for stepper motors. This method could also be applied to feedback from digital encoders.

1.3 **Structure of the Thesis**

Chapter Two describes the kinematics of the PUMA.

Chapter Three describes the algorithm and the splining procedure which was originally proposed in [2].

Chapter Four gives a detailed explanation of the proposed procedures and discusses the results of the simulation.

Chapter Five summarizes the findings of the thesis, states conclusions and offers directions for further research.

2 Kinematics of the PUMA 560 Robot

2.1 Introduction

2.1.1 Description of the PUMA 560

The PUMA 560 has six revolute joints as shown in figure 1. The Joint 1 axis coincides with the centerline of the trunk link l_1. The joint angle θ_1 is measured in the counter clockwise direction from the positive Y-axis.

The Joint axis 2 is perpendicular to and intersects the Joint 1 axis and coincides with the centerline of the shoulder. The shoulder is an offset of length d_1 between the trunk and upperarm. This offset is parallel to the X-Y plane and is in the negative X-axis direction when θ_1 is equal to zero. Link l_2, the upper arm, rotates about the Joint 2 axis an angle θ_2. The angle θ_2 is equal to zero when the link l_2 is parallel to the Z-axis as shown in figure 2a.

Joint 3, the elbow, has its axis parallel to the Joint 2 axis. The forearm, the third link, is formed of a two part link a and b shown in figure 3b. Link l_3 is the vector sum of a and b and is the distance between the Joint 3 axis and the center of the wrist. Link l_3 is offset from link l_2 by a distance d_2 parallel to the X-Y plane. When θ_1 is equal to zero, d_2 is parallel or in the positive X direction. When the arm is pointing up in the reference state(figure 2a), l_2 is parallel to the Z-axis and l_3 makes a known angle δ with the vertical. This angle is a function of the arm dimensions. Thus, at reference position, $\theta_{3r} = \delta$, where $\delta = \sin^{-1}(l_0/l_3)$. l_0 is defined as the offset

between the centerline of the two parts forming the forearm link l_3. Joints 4, 5, and 6

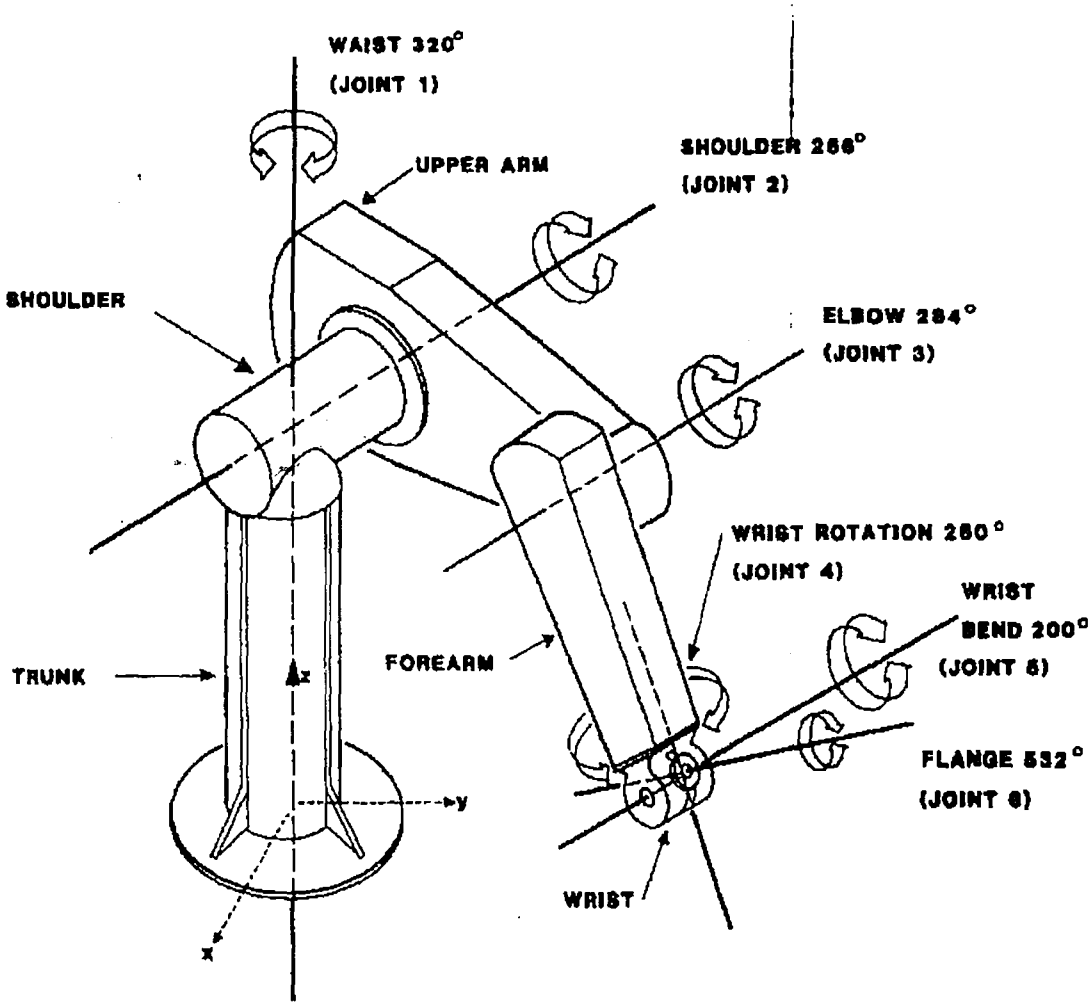


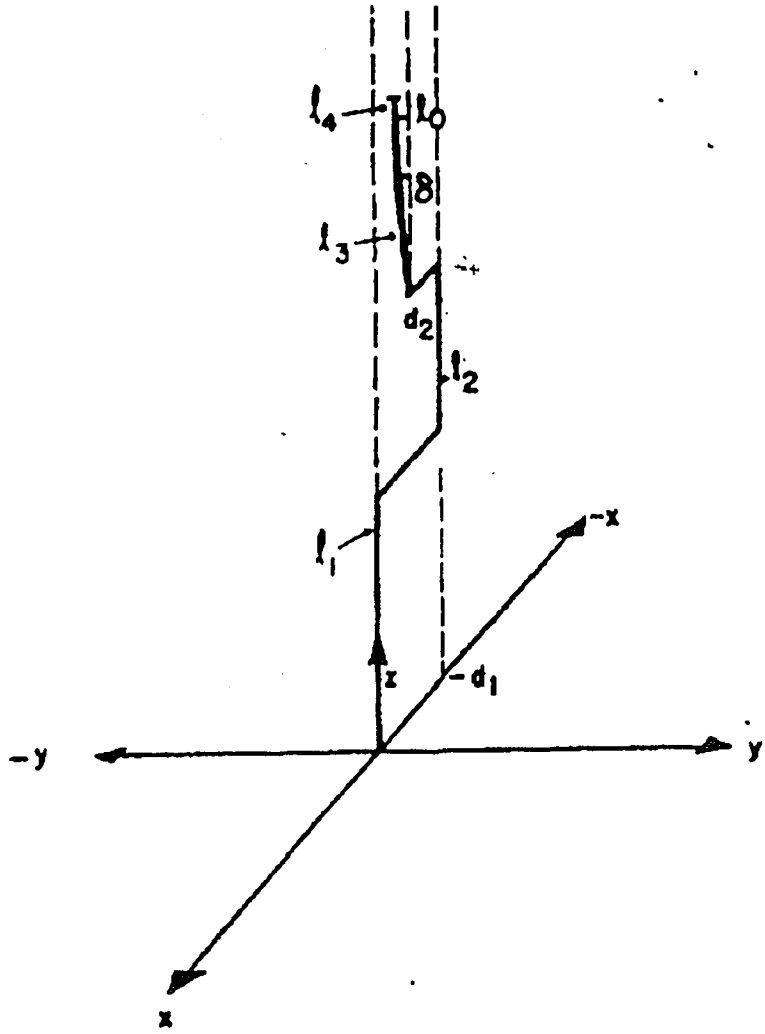
figure 1. PUMA 560. Degrees of rotation and member representation.[3]

form a spherical wrist. Joint 4 axis is perpendicular to and intersects the Joint 5 axis. Link l_4 is the link from the center of the wrist to the flange. Joint 4 rotates an angle θ_4 about its axis. Joint 5 axis is parallel to the axes of Joints 2 and 3. The angle θ_5 is the angle of rotation of link l_4 and is measured with respect to the Z-axis coordinates of link l_4, i.e. rotating the base coordinates through $\theta_1\mathbf{k}$, then through $-(\theta_2+\theta_3)\mathbf{i}$ and finally through $\theta_4\mathbf{k}$. The Joint 6 axis is perpendicular to and intersects the Joint 5 axis. It coincides with the centerline of the gripper mounting on the flange.

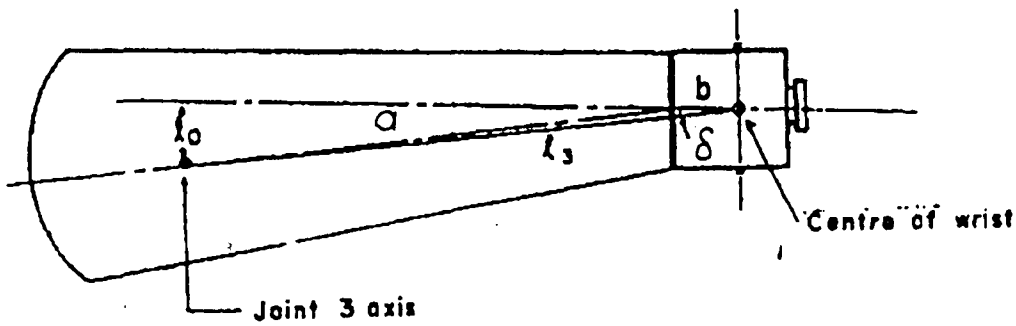
The position of the end effector in joint coordinates is expressed as $\theta = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)^T$ and in cartesian coordinates as $\mathbf{R} = (r_x, r_y, r_z, r_\rho, r_\theta, r_\mu)^T$ where the superscript T denotes the transpose. The position vector $\mathbf{r}$ is $(r_x, r_y, r_z)^T$ and r_ρ, r_θ, r_μ are the rotations about the Z-axis, the new negative X-axis, and the new Z-axis that aligns the base coordinates with the tip coordinates.

2.1.1.1 **Arm Configuration**

The robot arm has similarities with the human arm geometry. They are defined accordingly as having a shoulder, an elbow and a wrist. The robot arm may be lefty or righty, i.e it may resemble a human's left arm or right arm, respectively. The elbow can have two configurations: elbow up, where the elbow's position is above the line joining the shoulder and wrist, and elbow down, where the elbow is below that line.



(a)



(b)

figure 2. (a) Robot arm: Initial state. (b) Representation of link 1.[3]

The wrist has two configurations. The no-flip wrist for which θ_5 is positive, and the flip wrist where θ_5 is negative. Accordingly, the arm configuration parameters are defined as follows:

$k_1 = +1$ lefty

$k_1 = -1$ righty

$k_2 = +1$ elbow up

$k_2 = -1$ elbow down

$k_3 = +1$ no-flip wrist

$k_3 = -1$ flip wrist.

In the inverse kinematics case these parameters must be specified. In the direct case they can be computed.

2.1.1.2 **Singular Points**

Depending on the arm configuration there exists eight different solutions to the inverse kinematics. Singular points are the dividing points between the solution sets. These are the points where the Jacobian is zero and as a result there are several solutions to obtain the same end effector position. There are two important properties of singular points: (1) A loss in the number of degrees of freedom occurs, where the kinematic equations become less accurate in the neighborhood of these points and break down at the point itself. (2) The singular points are the points where the arm configuration changes.

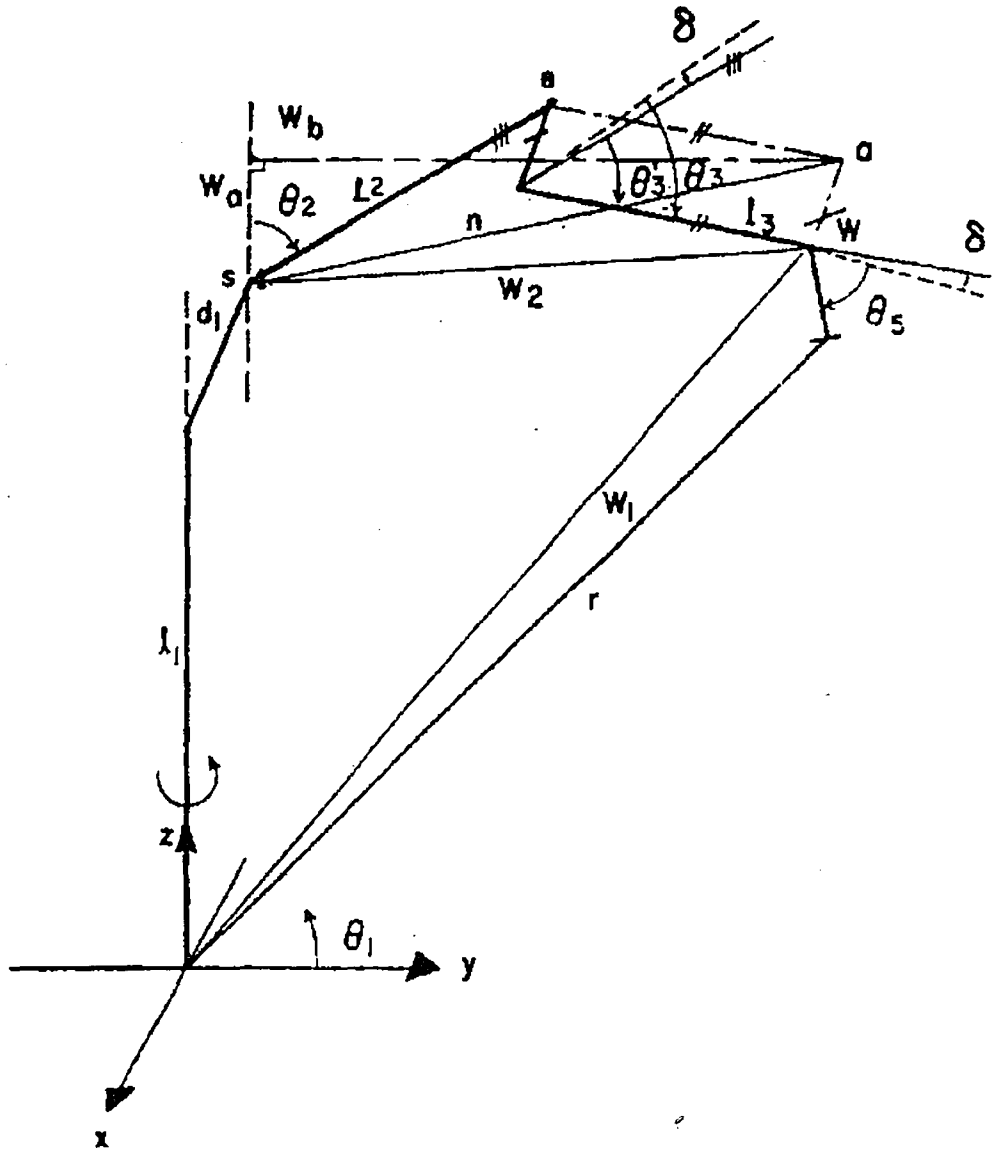


figure 3. Link representation of the PUMA 560.[3]

2.1.2 **Direct Kinematics**

Given the joint angles we would like to find out the arm configuration and the cartesian space coordinates.

2.1.2.1 **Orientation angles**

The orientation angles are computed using the spherical trigonometry formulas detailed by Featherstone[4] as follows:

$$\cos r_\theta = \cos(\theta_2+\theta_3)\cos\theta_5 - \sin(\theta_2+\theta_3)\sin\theta_5\cos\theta_4 \quad (1)$$

$$r_\mu = \theta_6 + \text{atan2}[\sin(\theta_2+\theta_3)\sin\theta_4,\ \sin\theta_5\cos(\theta_2+\theta_3)+\cos\theta_5\sin(\theta_2+\theta_3)\cos\theta_4] \quad (2)$$

$$r_\rho = \theta_1 + \text{atan2}[\sin\theta_5\sin\theta_4,\sin(\theta_2+\theta_3)\cos\theta_5 + \cos(\theta_2+\theta_3)\sin\theta_5\cos\theta_4] \quad (3)$$

where atan2(x,y) is the four quadrant version of $\tan^{-1}(x/y)$, which is used to avoid angle quadrant ambiguity inherent to trigonometry. As $|\cos r_\theta|$ goes to 1, the accuracy of the other two equations deteriorates. This is because $|\cos r_\theta| = 1$ is a singular point in the representations of rotations. If $\sin r_\theta = 0$, the orientation of the wrist will depend on the sum of r_ρ and r_μ and not on their individual values. In this analysis, if $\sin r_\theta = 0$, r_μ is set to zero and, r_θ is set to zero or π depending on the sign of $\cos r_\theta$. Then r_ρ is found using the equations developed in [8] as follows:

$$r_\rho = \theta_1 + \text{atan2}[2\sin(\theta_4+\theta_6)/\{\cos(\theta_2+\theta_3) + \cos r_\theta\},\cos(\theta_4+\theta_6)] \quad (4)$$

2.1.2.2 **Position Vector**

Referring to figure 4 and substituting θ_3' for $(\theta_3-\delta)$, the values of the projection of the vector **n** on the Z-axis w_a and on the X-Y plane w_b are expressed as

$$w_a = l_2\cos\theta_2 + l_3\cos(\theta_2+\theta_3')$$

$$w_b = l_2\sin\theta_2 + l_3\sin(\theta_2+\theta_3')$$

Since w_b makes an angle θ_1 with the Y-axis, the vectors **n**, $\mathbf{w_1}$, $\mathbf{w_2}$ can be written as

$$\mathbf{n} = -w_b\sin\theta_1\mathbf{i} + w_b\cos\theta_1\mathbf{j} + w_a\mathbf{k}$$

$$w_2 = (n_x + d_2\cos\theta_1)\mathbf{i} + (n_y + d_2\sin\theta_1)\mathbf{k} + w_a\mathbf{k}$$

$$w_1 = (-w_b\sin\theta_1 - d\cos\theta_1)\mathbf{i} + (w_b\cos\theta_1 - d\sin\theta_1)\mathbf{j} + (w_a+l_1)\mathbf{k}$$

where d is substituted for d_1-d_2. Since the vector $\mathbf{I}_4$ representing the link l_4 is

$$\mathbf{I}_4 = -l_4\sin r_\theta\sin r_\rho\mathbf{i} + l_4\sin r_\theta\cos r_\rho\mathbf{j} + l_4\cos r_\theta\mathbf{k},$$

the desired vector **r** is obtained by adding

$$r_x = -w_b\sin\theta_1 - d\cos\theta_1 - l_4\sin r_\theta\sin r_\rho \qquad (5)$$

$$r_y = w_b\cos\theta_1 - d\sin\theta_1 + l_4\sin r_\theta\cos r_\rho. \qquad (6)$$

$$r_z = w_a + l_1 + l_4\cos r_\theta. \qquad (7)$$

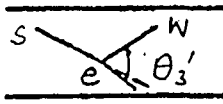
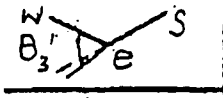
2.1.2.3 **Arm Configuration Parameters**

The value of k_1 depends on the arm being right or left. If the arm is a lefty, the projection of the arm on the X-Y plane, w_b, is positive, and vice versa for a right arm. Thus, to find k_1 the expression for w_b must be evaluated. If $w_b >= 0$, then the arm is a lefty and $k_1 = +1$. If $w_b < 0$, then the arm is a righty and $k_1 = -1$.

The value of k_2 depends on the elbow position. Table I shows four different combinations that might exist between k_1 and θ_3. It is concluded that if $k_1\theta_3 >= 0$, then the elbow is up and $k_2 = +1$. If $k_1\theta_3 < 0$, then the elbow is down and $k_2 = -1$.

The value of k_3 depends on the wrist. If $\theta_5 >= 0$, then a no flip solution exists and $k_3 = +1$. If $\theta_5 < 0$, then a flip solution exists and $k_3 = -1$.

Table I. Conditions on k_3 and θ_3 for elbow configuration.[3]

	KI	θ_3	ELBOW
	+1	>0	UP
	+1	<0	DOWN
	-1	>0	DOWN
	-1	<0	UP

2.1.3 **Inverse Kinematics**

Given the Cartesian vector **R** and the arm configuration parameters, we would like to find the joint angles vector θ.

Computation of θ_1.

The angle θ_1 is the result of a rotation of link l_1 around the Z-axis. To find θ_1, the position of the wrist w_1 with respect to the base coordinate is computed as the difference of the vectors **r** and $\mathbf{I}_4$

$$\mathbf{w}_1 = (r_x + l_4 \sin r_\theta \sin r_\rho)\mathbf{i} + (r_y - l_4 \sin r_\theta \cos r_\rho)\mathbf{j} + (r_z - l_4 \cos r_\theta)\mathbf{k}.$$

To obtain a closed form solution for θ_1, consider the projection l of the arm on the X-Y plane as shown in the figure 4. The projection of l_2 and l_3 on the X-Y plane is denoted as l_2' and l_3', respectively. Note that w_b is equal in magnitude to l. From figure 4b and figure 4c, θ_1 is calculated as

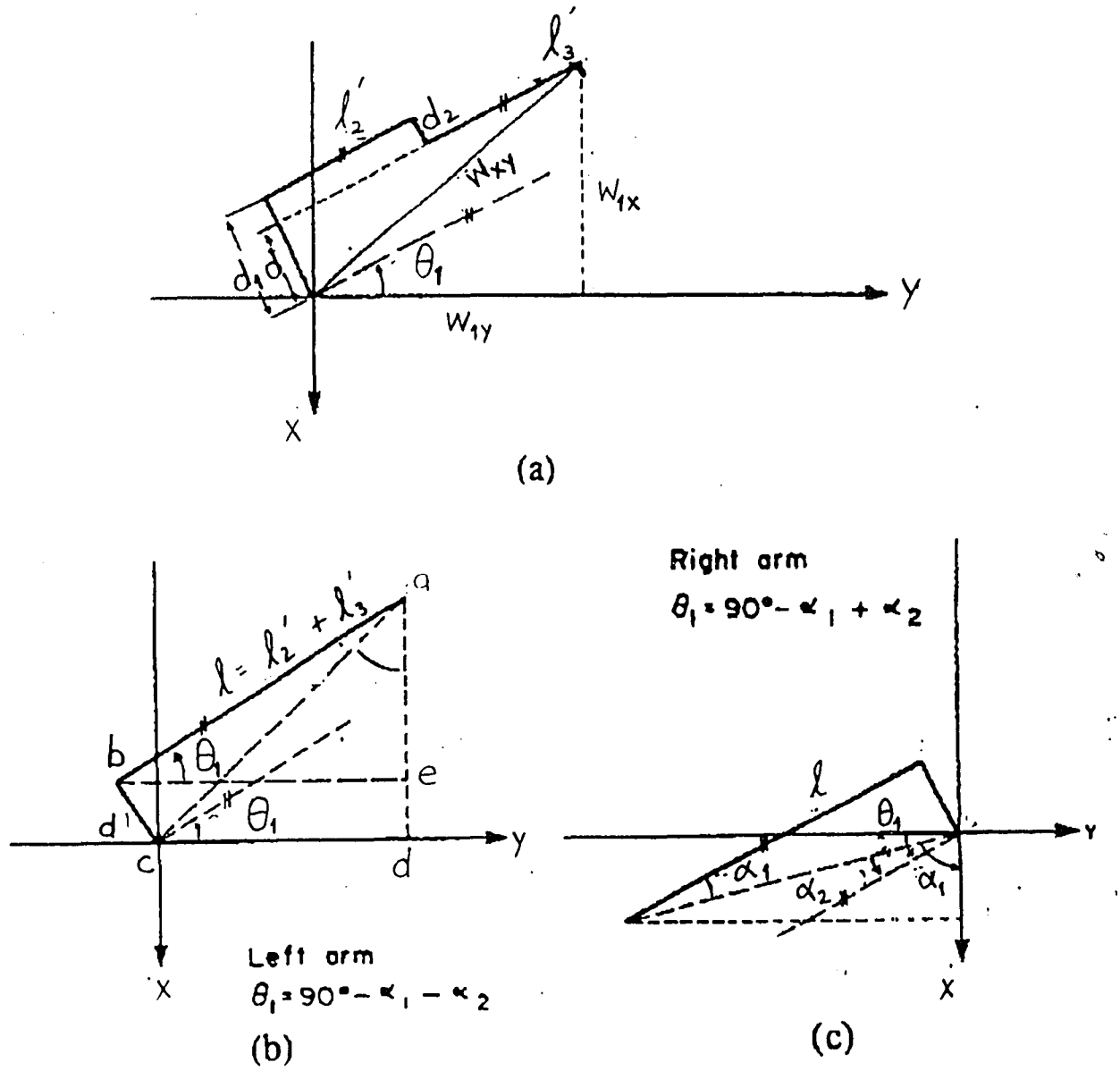


Figure 4. (a) Projection of the first three links on the X-Y plane. (b) Calculation of θ_1 for the left arm. (c) Calculation of θ_1 for the right arm.[3]

$$\theta_1 = 90 - \alpha_1 - k_1\alpha_2$$

where

$$\alpha_1 = \text{atan2}(k_1 w_{1y}, -k_1 w_{1x})$$

$$\alpha_2 = \text{atan2}(d,l),\ 0 \leq \alpha_2 \leq 90$$

and

$$l = (w^2_{xy} - d^2)^{1/2}.$$

Thus, θ_1 can be expressed as

$$\theta_1 = \text{atan2}(-k_1 w_{1x}, k_1 w_{1y}) - k_1 \text{atan2}(d,l). \quad (8)$$

Equation(8) indicates that a singular point exists if $w_{1x}=w_{1y}=0$. However, considering arm geometry, this condition is never satisfied.

Computation of θ_2 and θ_3

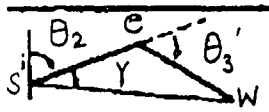
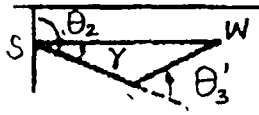
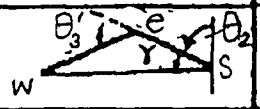
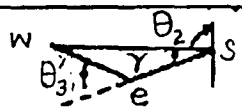
To find angles θ_2 and θ_3, consider figure 5 which represents links l_2 and l_3, the offset d_2, and the different angles used in the computation of θ_2 and θ_3. By the application of the cosine rule to s-e-w'

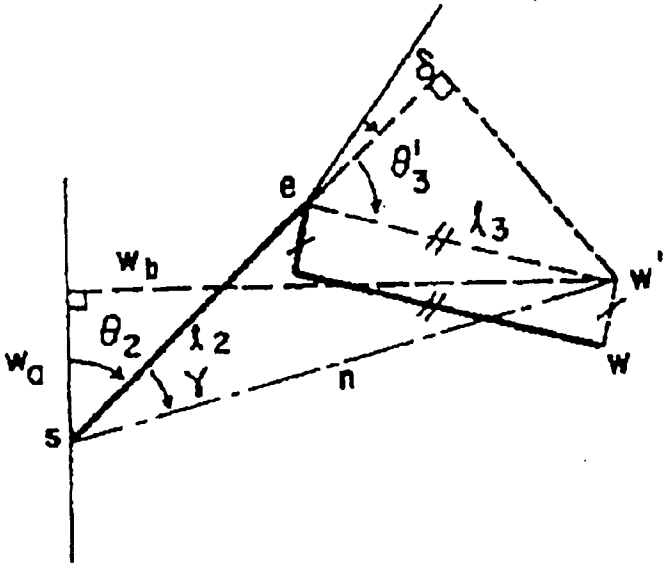
$$\cos\theta_3' = (n^2 - l^2_2 - l^2_3)/2l_2l_3 \quad (9)$$

If $|\cos\theta_3'| > 1$ then the position is unobtainable, and if $|\cos\theta_3'| = 1$, the manipulator is at a dead point.

Table II gives the different possible arm configurations for the first three links and the sign of the different angles. From the table, it is evident that for any arm configuration

Table II. Arm Configuration for the first three links and the signs of the angles involved.[3]

	K_1	K_2	K_1K_2	θ_2	θ_3'	γ
	+1	+1	+1	+ve	+ve	+ve
	+1	-1	-1	+ve	-ve	-ve
	-1	+1	-1	-ve	-ve	-ve
	-1	-1	+1	-ve	+ve	+ve



$$W_b^2 = n_x^2 + n_y^2$$

$$W_a' = W_{1z} - l_1$$

Figure 5. Calculation of θ_2 and θ_3 for left arm.[3]

$$\theta_3 = k_1 k_2 \theta_3' + \delta. \quad (10)$$

Also,

$$\theta_2 + \gamma = k_1 \text{atan2}(l, w_{1z} - l_1) \quad (11)$$

where

$$\gamma = k_1 k_2 \text{atan2}(l_3 \sin\theta_3', l_2 + l_3 \cos\theta_3').$$

Computation of θ_4, θ_5 and θ_6

To find these angles, Featherstone's equations[4] can be applied directly.

$$\theta_5 = \cos(\theta_2+\theta_3)\cos r_\theta + \sin(\theta_2+\theta_3)\sin r_\theta \cos(r_\rho - \theta_1) \quad (12)$$

If $|\cos\theta_5|$ is not equal to 1, $\sin\theta_5$ is computed as $(1-\cos^2\theta_5)^{1/2}$. If $\sin\theta_5 > \epsilon$, where ϵ is some small number, θ_4 and θ_6 are found from

$$\theta_4 = \text{atan2}[\sin r_\theta \sin(r_\rho - \theta_1), \cos(\theta_2+\theta_3)\sin r_\theta \cos(r_\rho - \theta_1) - \sin(\theta_2+\theta_3)\cos r_\theta] \quad (13)$$

$$\theta_6 = r_\mu - ß \quad (14)$$

where

$$ß = \text{atan2}[\sin(\theta_2+\theta_3)\sin(r_\rho - \theta_1), \sin r_\theta \cos(\theta_2+\theta_3) - \cos r_\theta \sin(\theta_2+\theta_3)\cos(r_\rho - \theta_1)]$$

The manipulator loses a degree of freedom whenever two joint axes become colinear. This is the case when $\sin\theta_5 = 0$ and consequently θ_4 and θ_6 become linearly dependent[4].

The accuracy of θ_4 and ß deteriorate as $\sin\theta_5 \to 0$, and they break down completely if $\sin\theta_5 = 0$. Therefore, for some value of ϵ and for $|\sin\theta_5| < \epsilon$, better correspondence can be obtained between θ_4 and ß by using the equation

$$\theta_4 - ß = \text{atan2}[(1/2)\sin(r_\rho - \theta_1)\{\cos(\theta_2+\theta_3) + \cos r_\theta\}, \cos(r_\rho - \theta_1)] \quad (15)$$

In case of a no-flip condition, that is $k_3 = 1$, the wrist angles are obtained from the above equations. If however, $k_3 = -1$, the flip solution becomes $(\theta_4+\pi, -\theta_5, \theta_6+\pi)$.

3.1 Cartesian Path Approximation

In this chapter three procedures are proposed for selecting intermediate knot points along an arbitrary smooth cartesian path. The transformation of these points from cartesian to joint space is also dealt with.

Let the position path of an arbitrary smooth path be represented by **P(t)**, where **P(t)** is a 3 x 1 position vector. The curvature **k(t)** can be calculated as follows[2].

$$k(t) = ((\dot{\mathbf{P}}\cdot\dot{\mathbf{P}})(\ddot{\mathbf{P}}\cdot\ddot{\mathbf{P}})-(\dot{\mathbf{P}}\cdot\ddot{\mathbf{P}})^2)^{1/2} / (\dot{\mathbf{P}}\cdot\dot{\mathbf{P}})^{3/2} \qquad (16)$$

where the dot denotes the inner vector product or dot product. The curvatures corresponding to adjacent points on a smooth path differ only slightly. If the difference in curvatures is less than a prescribed tolerance, then the path between the two points can be considered to be approximately circular with a constant curvature. A path segment is then obtained between two points with approximately the same curvature. Based on the position trajectory of the given cartesian path, the path segments are successively determined. The orientation along the path is also known and can be paramterized by 't.' The orientation corresponding to the two endpoints is obtained, after the path segment has been determined.

The second procedure uses concatenated line segments to approximate each path segment obtained from the previous procedure. Since each path segment is considered to be approximately circular, the problem is reduced to approximating a

circle by a polygon. The ratio of the area between the circle and an N sided polygon should ideally be 1. But since the polygon is only an approximation of the circle,

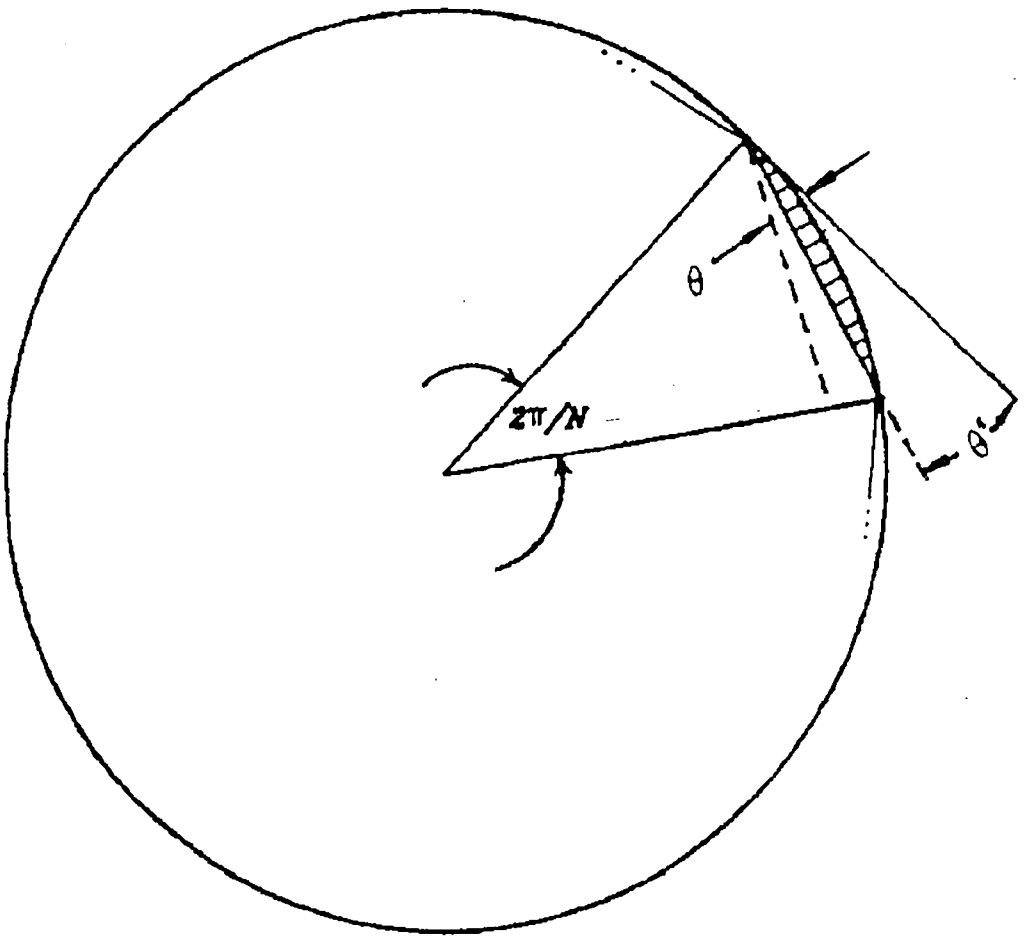


figure 6. The approximate error area between a polygon and a circle.[3]

(dotted line represents a non optimal side)

there is an error in this ratio and this error is specified to be less than a specified tolerance ϵ. The relationship between N and ϵ is

$$2\pi(1-\epsilon) = N\sin(2\pi/N) \qquad (17)$$

For a given ϵ, the smallest N that satisfies (17) is designated as N^*. Let the angle between the line tangent to the circle at one vertex of the polygon and the polygon side be denoted as θ (see figure 6). Then θ is approximately equal to π/N. Let θ^* be denoted by π/N^*, where θ^* can be considered as a reference index to decide whether the approximation meets the specified tolerance or not. Consider the approximation of a path by concatenated line segments. If the angle θ between the tangent line at one point on the path and the line segment extending from that point is larger than the reference angle θ^*, then the line segment is not the desired one (see figure 6). An iterative procedure is carried out until a line segment which has a θ smaller than θ^* is found. The position and orientation corresponding to the two endpoints of the line, are saved. This procedure is repeated until all the path segments are approximated by line segments. Note that the endpoints of all line segments lie on the cartesian path itself.

Once the concatenated line segments are determined, the next step is to determine intermediate knot points which are subsequently transformed to joint space. The procedure is similar to the one used in [9] where the positions and orientations corresponding to the endpoints of the line segments are utilized. The endpoints of a

line segment are transformed to joint space. The mid point between these two points in joint space is determined and transformed back to cartesian space. This transformed point is compared with the mid point of the line segment calculated from the cartesian space endpoints. The deviation between these two points in position and orientation should be within a prescribed tolerance. If not, the mid point in cartesian space is chosen as the end point of the same line segment and the procedure is repeated until the deviations are within specified bounds. Once an intermediate knot point has been selected, this procedure is repeated between the knot point and the original endpoint until a series of intermediate knot points is selected for a single line segment. This procedure is then repeated for successive line segments, finally resulting in a series of knot points describing the cartesian path.

The knot points must now be transformed to joint space. For this purpose, the equations developed in [3] were utilized. The solution is based on a method that fully exploits the special geometry of the PUMA 560 robot. Special attention is given to the arm configuration in both directions. The algorithm presented here is an adaptation of the one presented in [2].

3.2 **Divide a Cartesian Path into Path Segments.**

The motion of the end effector in cartesian space is expressed as

$$\mathbf{H} = [\ P_x,\ P_y,\ P_z,\ \rho_x,\ \rho_y,\ \rho_z\ ]^T = [\mathbf{P}^T,\ \mathbf{T}^T]^T$$

where

$\mathbf{P}^T = [P_x, P_y, P_z]$ is the position vector,

$\mathbf{R}^T = [\rho_x, \rho_y, \rho_z]$ is the orientation vector.

The angles ρ_x, ρ_y, ρ_z are the rotational angles about the Z-axis, the new X-axis and the new Z-axis respectively, that align the manipulator base coordinates with the end effector coordinates.

The algorithm to divide the cartesian path into segments is as follows:

1) Let Ts be the starting time and Tg be the ending time of the cartesian path. Let i denote the subscript for the ith path segment. Assign To = Ts and Tf = To.(Start at the beginning of the path)

2) Compute δ_t = (Tg - Ts)/df where df usually is 2. Thus δ_t is an increment in time. Compute Tf = To + $n\delta_t$ where n starts from 0. 'n' denotes the number of time increments advanced.

Check whether $|k(To) - k(Tf)| > \xi_1$ (18)

where k(T) is the curvature of the path at time T and is expressed using equation (16).

If (18) is not true, n = n+1 until (18) is satisfied.

If Tf >= Tg then set Tf = Tg and end this procedure. Otherwise, go to the next step.

In this step we are trying to find a range of time in which the curvature of the path is approximately constant. We move forward in time starting with Tf = To. Thus, for

example the above condition may be satisfied when Tf is To + $3\delta_t$. In this case the actual value of Tf when the difference in curvatures exceeds ξ_1 is somewhere between To + $2\delta_t$ and To + $3\delta_t$.

3) Now let $\delta_t' = \delta_t/df$. Let m = 0;

$$Tf = Tf - m\delta_t' \tag{19}$$

check whether $| k(To)-k(Tf) | < \xi_1$ (20)

If (20) not true, m = m+1 and repeat eqns (19) and (20) until the above condition is satisfied.

The algorithm as stated above and in [2] will not work in this case because equation (20) will eventually give a negative value for Tf when equations (20) and (21) are repeated. In order to overcome this problem, let $Tf' = Tf - m\delta_t'$ and check whether $| k(To)-k(Tf') | < \xi_1$. Increment m until the condition is satisfied. If Tf' equals To then let Tf' be equal to the original Tf. Now we are moving back in time in steps of δ_t' in order to determine the path segment whose curvature is approximately constant.

4) Save the endpoints of the segment. Hi = H(Tf'). The endpoints of the segment in time are now Tsi = To and Tgi = Tf'. Let To = Tf' and repeat steps 2, 3 and 4.

5) End of procedure.

3.3 **Approximating Path Segments with Line segments**

In the previous procedure the cartesian path was split into segments so that each segment can be considered as a circular arc. In this procedure these path segments are approximated by line segments. The procedure for approximating a path segment with straight line segments is as follows:

1) Let To = Tsi and Tf = Tgi starting with i = 0;

The unit tangent vector T(To) at H(To) is given by

$\mathbf{T}(To) = \mathbf{P}(To)/ \mid \mid \mathbf{P}(To) \mid \mid = [t_x(To), t_y(To), t_z(To)]^T$

where the operator $\mid \mid \mathbf{I} \mid \mid$ denotes the magnitude of the vector **I**.

2) The unit vector from H(To) to H(Tf) is given by

$\mathbf{L}(To,Tf) = (\mathbf{P}(Tf)-\mathbf{P}(To))/ \mid \mid \mathbf{P}(Tf)-\mathbf{P}(To) \mid \mid$

Let $\mathbf{L}(To,Tf) = [l_x(To,Tf), l_y(To,Tf), l_z(To,Tf)]^T$

The angle between the tangent vector and the unit line vector is given by

$\Theta = ATAN2(1-(t_xl_x + t_yl_y + t_zl_z), t_xl_x + t_yl_y + t_zl_z)$

This formula, given in [2] is incorrect. The correct formula is

$\Theta = ACOS(1-(t_xl_x + t_yl_y + t_zl_z), t_xl_x + t_yl_y + t_zl_z)$

From the first procedure the path segments of same curvature were found. As explained previously, approximating the path segments by line segments is done by first considering the approximation of a circle by an N sided polygon. The condition

to be fulfilled here is that the ratio of the area of the polygon to that of the circle should be less than or equal to a specified tolerance ξ_2. Thus we get

$$2\pi(1-\xi_2) = N\sin(2\pi/N)$$

For a given ξ_2 the smallest integer N that satisfies this equation is designated as N*. The angle between the tangent line at one vertex of the polygon and the corresponding polygon side is given by π/N. Thus $\Theta^* = \pi/N^*$. This can be considered as the optimum angle. If the Θ that is calculated is greater than Θ^*, then N is not the required number of line segments and the iterative procedure that follows is used for calculating N.

3) Check whether $| \theta^* - \theta | < \xi_2$

If this is not satisfied, then let Tf = (To+Tf)/df and go to the previous step 2.

Thus, we go back in time until we obtain a line segment that approximates that part of the cartesian path between To and Tf.

4) Save the endpoints of the line segments, i.e. $H_j(Tf)$. Let To = Tf, Tf = Tgi and increment j by using j = j+1. Repeat these four steps until To = Tgi. Then do the same procedure for the next path segment and so on.

3.4 **Generation of Intermediate Knot Points.**

In general, when a curve is to be approximated by straight lines, it is desirable to sample as many points as possible on the curve. But since this is computationally ineffecient, points on the curve are selected based on approximations. This procedure describes the selecting of intermediate knot points on the cartesian path segment.

1) Consider the first line segment of the first path segment, or generally speaking the jth line segment of the ith path segment.

Let $\mathbf{H}_{ij} = \mathbf{H}_{left}$ and $\mathbf{H}_{i\,j+1} = \mathbf{H}_{right}$

2) Find the joint vectors corresponding to the above two positions. Let these be called $\mathbf{J}_{left}$ and $\mathbf{J}_{right}$

In transforming coordinates from cartesian space to joint space we make use of the equations given in [3] and [4]. These equations have been specially developed for the PUMA 560 taking into account the specific geometry and possible configurations for a single coordinate. The derivation of these equations is explained in [4] and these have been modified to specially suit the PUMA 560 in [3].

Compute the mid point in joint space by

$\mathbf{J}_{mid} = (\mathbf{J}_{right} + \mathbf{J}_{left})/2$

Now the corresponding transformation to cartesian space is done by forward kinematics on $\mathbf{J}_{mid}$ giving $\mathbf{H}_{mid} = [\mathbf{P}^T_{mid}, \mathbf{R}_{mid}{}^T]^T$.

Also, compute the mid point in cartesian space

$$\mathbf{H}_{center} = (\mathbf{H}_{right} + \mathbf{H}_{left})/2 = [\mathbf{P}^T_{center}, \mathbf{R}^T_{center}]^T$$

4) The deviations between the mid point calculated from the cartesian space coordinates and the mid point in cartesian space computed by the forward kinematics of the mid point in joint space are calculated as

$$\delta_p = \Sigma \mid \mathbf{P}_{mid} - \mathbf{P}_{center} \mid$$

$$\delta_r = \Sigma \mid \mathbf{R}_{mid} - \mathbf{R}_{center} \mid$$

5) Check whether $\delta_p < \delta_{p\,max}$ and $\delta_r < \delta_{r\,max}$, where $\delta_{p\,max}$ and $\delta_{r\,max}$ have been previously specified by the user. If these conditions are not satisfied, then let $\mathbf{H}_{right} = \mathbf{H}_{center}$ and repeat the steps 2 to 4. until the conditions are satisfied, i.e. move back in space along the line segment until a point can be selected so that that segment in joint space is a reasonable approximation of the corresponding line segment. If the conditions are never satisfied, then take the mid point of the original line segment as a knot point and move on to the next line segment. Let $\mathbf{H}_{left} = \mathbf{H}_{right}$ and $\mathbf{H}_{right} = \mathbf{H}_{goal}$ where $\mathbf{H}_{goal} = \mathbf{H}_{i\,j+1}$. This procedure is repeated for each and every line segment, resulting in a series of points in joint space which have to be splined.

3.5 **Splining of Knot Points.**

As a result of the previous procedures, a series of knot points in joint space has been obtained. As explained previously, when these points are tranformed back to cartesian space they will approximate the given cartesian path within specified error bounds. In order to obtain a continuous trajectory in joint space, these knot points must be splined. The procedure for splining developed in [2] is explained first. This is followed by an explanation of the procedure developed by the author in the next chapter.

Let the knot points in space be arranged as $y(t_0)$, $y(t_1)$ ...$y(t_n)$, where $y(t_0)$ denotes the knot point 'y' at time t_0. A fourth order polynomial $Q_i(t)$ can be represented as

$$Q_i(t) = a_i(t-t_i)^4 + b_i(t-t_i)^3 + c_i(t-t_i)^2 + d_i(t-t_i) + e_i \qquad (21)$$

where $Q_i(t)$ is the spline joining the points $y(t_i)$ and $y(t_{i+1})$ and is parameterized by the time variable $t \in [t_i, t_{i+1}]$ and $i = 0, 1, 2,\ldots, n-1$. The path continuity constraints are

$$Q_i(t_i) = Q_{i-1}(t_i),\ Q_i(t_i) = Q_{i-1}(t_i),\ Q_i(t_i) = Q_{i-1}(t_i)$$

Further, for local smoothness it is required that

$\int_{t_i}^{t_{i+1}} [Q_i(t)]^2 dt$ be minimized.

Based on the above criterion recursive formulas for calculating the coeffecients in the above equation are as follows.

$$e_i = y_i \tag{22}$$

$$d_i = 4a_{i-1}h^3_{i-1} + 3b_{i-1}h^2_{i-1} + 2c_{i-1}h_{i-1} + d_{i-1} \tag{23}$$

$$c_i = 6a_{i-1}h^2_{i-1} + 3b_{i-1}h_{i-1} + c_{i-1} \tag{24}$$

$$b_i = (9/4)(y_{i+1} - y_i)h^{-3}_i - (9/4)d_ih^{-2}_i - (11/6)c_ih^{-1}_i \tag{25}$$

$$a_i = -(5/4)(y_{i+1} - y_i)h^{-4}_i + (5/4)d_ih^{-3}_i + (5/6)c_ih^{-2} \tag{26}$$

where $h_i = t_{i+1} - t_i$ for i = 2,3,...,n-3. The variable 'h' is used to denote the difference in time between two knot points. The knot points used in these equations are y_i and y_{i+1} where y_i represents $y(t_i)$. This is done for the sake of legibility.

The initial values for the equations (22-26) are c_2 and d_2. Let tracking error be defined as

$$TE_i = {}_{ti}^{ti+1}\int [Q_i(t) - L_i(t)]^2 \, dt$$

where $L_i(t)$ = desired path between y_i and y_{i+1} and is given by

$$L_i(t) = y_i + (y_{i+1} - y_i)(t_{i+1} - t_i)^{-1}(t-t_i)$$

The total tracking error is Σ TE_i, i = 2 to n-3.

To simplify computation, c_2 is taken as constant. Thus, this becomes a single variable optimization problem. $\partial TE/\partial d_2$ yields

$$d_2^* = (y_3-y_2)h^{-1} + (k_2/k_1)c_2h_2$$

where k_2 = 2416 and k_1 = -1057.2

In other words d_2 can be taken to be the user specified initial velocity, and c_2 can be the user specified acceleration. The final segments must satisfy the following constraints:

$$Q_{n-2}(t_{n-1}) = Q_{n-1}(t_{n-1})$$

$$\dot{Q}_{n-2}(t_{n-i}) = \dot{Q}_{n-1}(t_{n-1})$$

$$\ddot{Q}_{n-2}(t_{n-1}) = \ddot{Q}_{n-1}(t_{n-1})$$

$$Q_{n-2}(t_{n-1}) = y_{n-2}$$

$$\dot{Q}_{n-2}(t_{n-1}) = \omega_{n-2}$$

$$\ddot{Q}_{n-2}(t_{n-1}) = \alpha_{n-2}$$

$$Q_{n-1}(t_n) = y_{n-1,}, \; Q_{n-2}(t_n) = \omega_{n-1}, \; Q_{n-2}(t_n) = \alpha_n$$

where ω_{n-2}, ω_{n-1}, α_{n-2} and α_n are the desired accelerations and velocities.

Since there are more equations than unknowns, the solutions will not be unique. Hence a pseudo knot point is added. This point is chosen so that the tracking error in the final segments is minimized. The values of the coeffecients for the pseudo point are found from

$$M_f X_f = N_f$$

where

$$M_f = \begin{bmatrix} h^4 & h^3 & 0 & 0 & 0 & 0 & -1 \\ 4h^3 & 3h^2 & 0 & 0 & 0 & -1 & 0 \\ 12h^2 & 6h & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & h^4 & h^3 & h^2 & h & 1 \\ 0 & 0 & 4h^3 & 3h^2 & 2h & 1 & 0 \\ 0 & 0 & 12h^2 & 6h & 2 & 0 & 0 \\ \frac{15h^4}{56} & \frac{17h^3}{72} & h^4/5 & h^3/4 & h^2/3 & h/2 & \frac{379}{252} \end{bmatrix} \qquad (27)$$

$$X_f = \begin{bmatrix} a_{n-2} \\ b_{n-2} \\ a_{n-1} \\ b_{n-1} \\ c_{n-1} \\ d_{n-1} \\ e_{n-1} \end{bmatrix} \tag{28}$$

$$N_f = \begin{bmatrix} e_{n-2} - d_{n-2}h - c_{n-2}h^2 \\ - d_{n-2} - c_{n-2}h \\ -2c_{n-2} \\ e_n \\ \omega_n \\ \alpha_n \\ \eta_n \end{bmatrix} \tag{29}$$

where $\eta_n = (719/1260)e_{n-2} + (14/15)e_n + (7/36)c_{n-2}h^2 + (173/1260)d_{n-2}h$

and $h = (1/2)(t_n - t_{n-2})$

In this case the end velocity and acceleration are user specified.

The pseudo knot y_1 is likewise added at the beginning. The coeffecients for this are found from

$M_sX_s = N_s$ where $M_s = M_f$ and

$$X_s = \begin{bmatrix} a_0 \\ b_0 \\ a_1 \\ b_1 \\ c_1 \\ d_1 \\ e_1 \end{bmatrix} \tag{30}$$

$$N_s = \begin{bmatrix} e_0 - d_0h - c_0h^2 \\ - d_0 - c_0h \\ -2c_0 \\ e_2 \\ \omega_2 \\ \alpha_2 \\ \eta_0 \end{bmatrix} \tag{31}$$

$$h = (1/2)(t_2 - t_0)$$

$$\eta_0 = (719/1260)e_0 + (14/15)e_2 + (7/36)c_0h^2 + (173/1260)d_0h$$

Now the coordinates of a single joint have been splined. This procedure is repeated for all the other joints.

4.1 Proposed Procedure.

Two methods are proposed in the following sections. The same smoothness criterion is used in developing the recursive equations to calculate the spline coeffecients. In this case the starting values for the system of equations are c_0, d_0 and e_0, the initial acceleration, velocity and position of the end effector, respectively. Substituting these in the recursive equations for $t = t_0$ to t_{n-1} will give us the values of the spline coeffecients, but the final values of acceleration and velocity will be determined by the equations and not by the user. In order to allow the user to specify a final velocity and acceleration, the following is procedure is adopted.

4.1.1 Procedure 1

At time t_{n-2} the values of c_{n-2}, d_{n-2}, e_{n-2} are known. Instead of using these values to find a_{n-2} and b_{n-2} in the recursive equations, let a_{n-2}, the fourth order coeffecient in the spline Q_{n-2} be denoted as a_{n-2} itself. As explained previously, if the recursive equations are used to compute all the coeffecients, the end velocity and acceleration becomes determined by the system of equations itself and not by the user. Substitute for b_{n-2} in terms of a_{n-2} in the equation for the spline Q_{n-2} at time t_{n-1}.

$$Q_{n-2} = a_{n-2}h^4 + b_{n-2}h^3 + c_{n-2}h^2 + d_{n-2}h + e_{n-2}$$

where

$$h = t_{n-1} - t_{n-2}$$

At time t_{n-1}

$$Q_{n-1} = e_{n-1}$$

By the constraint of continuity, the above two equations are equal.

Hence,

$$b_{n-2} = (1/h^3)(e_{n-1} - e_{n-2} - d_{n-2}h - c_{n-2}h^2 - a_{n-2}h^4)$$

The acceleration constraint gives us

$$c_{n-1} = 6a_{n-2}h^2 + 3b_{n-2}h + c_{n-2}.$$

Since b_{n-2} is already known in terms of a_{n-2}, now c_{n-1} is also in terms of a_{n-2}.

The velocity constraint gives us

$$d_{n-1} = 4a_{n-2}h^3 + 3b_{n-2}h^2 + 2c_{n-2}h + d_{n-2}.$$

Therefore d_{n-1} is also now in terms of a_{n-2}.

Let a_{n-1}, the fourth order coeffecient of the spline Q_{n-1} be denoted as a_{n-1}.

At time t_n

$$Q_{n-1} = a_{n-1}h^4 + b_{n-1}h^3 + c_{n-1}h^2 + d_{n-1}h + e_{n-1}.$$

$$Q_n = e_n$$

where

$$h = t_n - t_{n-1}$$

Therefore b_{n-1} in terms of a_{n-1} is

$$b_{n-1} = (1/h^3)(e_n - e_{n-1} - d_{n-1}h - c_{n-1}h^2 - a_{n-1}h^4) \qquad (32)$$

c_{n-1} and d_{n-1} are already in terms of a_{n-2}. So b_{n-1} is now in terms of a_{n-1} and a_{n-2}.

The acceleration constraint gives us

$$c_n = 6a_{n-1}h^2 + 3b_{n-1}h + c_{n-1}. \qquad (33)$$

Since b_{n-1} is already known in terms of a_{n-1} and a_{n-2}, now c_n is also in terms of a_{n-1} and a_{n-2}. The velocity constraint gives us

$$d_n = 4a_{n-1}h^3 + 3b_{n-1}h^2 + 2c_{n-1}h + d_{n-1}. \quad (34)$$

Therefore d_n is also now in terms of a_{n-1} and a_{n-2}. The final acceleration and velocity are c_n and d_n, respectively. They are now user specified. Once c_n and d_n are specified, equations (33) and (34) can be solved for a_{n-1} and a_{n-2}. From these values the rest of the coeffecients can be found since they are in terms of a_{n-1} and a_{n-2}.

4.1.2 **Procedure 2**

In this procedure the recursive equations are used instead of the equation of the spline itself, to find the coeffecients of all the splines. Two pseudo knot points are added to the original set of points. The first pseudo knot is added between the knot points n-2 and n-1 and is assumed to be at time

$$t' = t_{n-2} + ((t_{n-1} - t_{n-2})/2).$$

Let the coeffecients of the spline at the first pseudo knot be a', b', c', d', e'. Let the coeffecients at the second pseudo point be a", b", c", d", e". The second pseudo point is added between the knot points at n-1 and n at time

$$t'' = t_{n-1} + ((t_n - t_{n-1})/2).$$

At time t_{n-2}, c_{n-2} and d_{n-2} are known from the recursive equations. In the equation for a_{n-2}, substitute y' instead of y_{i+1}.

$$a_{n-2} = -(5/4)(y' - y_i)h^{-4}_i + (5/4)d_{n-2}h^{-3}_{n-2} + (5/6)c_{n-2}h^{-2}_{n-2} \quad (35)$$

$$b_{n-2} = (9/4)(y' - y_{n-2})h^{-3}_{n-2} - (9/4)d_{n-2}h^{-2}_{n-2} - (11/6)c_{n-2}h^{-1}_{n-2} \quad (36)$$

where

$$h_{n-2} = t' - t_{n-2}$$

These values of a_{n-2} and b_{n-2} are substituted in the equations for c' and d'

$$c' = 6a_{n-2}h^2_{n-2} + 3b_{n-2}h_{n-2} + c_{n-2} \quad (37)$$

$$d' = 4a_{n-2}h^3_{n-2} + 3b_{n-2}h^2_{n-2} + 2c_{n-2} + d_{n-2} \quad (38)$$

$$e' = y' \quad (39)$$

Since a_{n-2} and b_{n-2} are in terms of y', c' and d' are also now in terms of y'.

Substituting these values in the recursive equations we get

$$a' = -(5/4)(y_{n-1} - y')h'^{-4} + (5/4)d'h'^{-3} + (5/6)c'h'^{-2} \quad (40)$$

$$b' = (9/4)(y_{n-1} - y')h'^{-3} - (9/4)d'h'^{-2} - (11/6)c'h'^{-1} \quad (41)$$

$$c_{n-1} = 6a'h'^2 + 3b'h' + c' \quad (42)$$

$$d_{n-1} = 4a'h'^3 + 3b'h'^2 + 2c'h' + d' \quad (43)$$

$$e_{n-1} = y_{n-1} \quad (44)$$

where

$$h' = t_{n-1} - t'$$

c_{n-1} and d_{n-1} are now in terms of y'.

$$a_{n-1} = -(5/4)(y'' - y_{n-1})h^{-4}_{n-1} + (5/4)d_{n-1}h^{-3}_{n-1} + (5/6)c_{n-1}h^{-2}_{n-1} \quad (45)$$

$$b_{n-1} = (9/4)(y'' - y_{n-1})h^{-3}_{n-1} - (9/4)d_{n-1}h^{-2}_{n-1} - (11/6)c_{n-1}h^{-1}_{n-1} \quad (46)$$

$$c'' = 6a_{n-1}h^2_{n-1} + 3b_{n-1}h + c_{n-1} \quad (47)$$

$$d'' = 4a_{n-1}h^3_{n-1} + 3b_{n-1}h^2_{n-1} + 2c_{n-1}h + d_{n-1} \quad (48)$$

$$e'' = y'' \quad (49)$$

where

$$h_{n-1} = t" - t_{n-1}. \tag{50}$$

Since c_{n-1} and d_{n-1} are in terms of y', a_{n-1} and b_{n-1} are in terms of y" and y'. Thus c" and d" are also now in terms of y' and y". Substituting these values in the recursive equations again we get

$$a" = -(5/4)(y_n - y")h"^{-4} + (5/4)d"h"^{-3} + (5/6)c"h"^{-2} \tag{51}$$

$$b" = (9/4)(y_n - y")h"^{-3} - (9/4)d"h"^{-2} - (11/6)c"h"^{-1} \tag{52}$$

$$c_n = 6a"h"^2 + 3b"h" + c" \tag{53}$$

$$d_n = 4a"h"^3 + 3b"h"^2 + 2c"h" + d" \tag{54}$$

$$e_n = y_n \tag{55}$$

where

$$h" = t_n - t".$$

The final acceleration and velocity are c_n and d_n, respectively. They are user specified. Once these are specified, equatons (54) and (55) can be solved for y' and y". The values of y' and y" so obtained can be substituted back into the equations above to determine the coeffecients a_{n-2}, b_{n-2}, a', b', c', d', e', a_{n-1}, b_{n-1}, c_{n-1}, d_{n-1}, e_{n-1}, a", b", c", d", and e". Now that the points have been splined, the paths have to be tracked.

4.2 **Path Tracking in Joint Space.**

The PUMA 560 is assumed to have a micro stepping motor. Consider a single joint. The position of the shaft is indicated by sensors. This is compared to the required position calculated from the corresponding spline equation. The square of the distance between the two is calculated. A step is taken in the direction that reduces the square of the distance between the two positions. The basis for this algorithm was obtained from [6].

1) Find present position of the shaft from sensors at time t

2) The position of the shaft at time t+ (time it takes for the shaft to move through a step) is calculated from the corresponding spline equation.

3) The distance moved is divided by the resolution of the stepper motor in order to obtain the number of pulses.

4) The position of the shaft +

a) (# of pulses*resolution) in one direction

b) (# of pulses*resolution) in the opposite direction

c) no step

is found. The squares of the distance of each of the three cases from the position calculated in step 2 is found. The option that gives the minimum of these squares is chosen. This tracking algorithm is applied to the splines for each joint.

4.3 **Parameters of the Model used for the Simulation**

A simulation was performed in order to test the proposed procedures. The simulation program was written in C and executed on an IBM RISC 6000 workstation. The cartesian path, configuration of the PUMA 560 and stepper motor parameters that were used in the simulation are given in the following sections.

4.3.1 **Cartesian path**

The end effector of the robot is to follow a cartesian path that is parameterized by the time variable 't' and is defined by

$$\mathbf{H}(t) = [P_x(t), P_y(t), P_z(t), \rho_x(t), \rho_y(t), \rho_z(t)]^T$$

where

$$P_x(t) = \theta(t) - \sin\theta(t),\ P_y(t) = 1 - \cos\theta(t),\ P_z(t) = 0$$

$$\rho_x(t) = \pi/2 - \pi t/4,\ \rho_y(t) = \pi/2,\ \rho_z(t) = \pi/2$$

$$\theta(t) = 2\pi t \text{ and } 0.01 \leq t \leq 1.$$

The starting and ending positions in terms of base coordinates are given, respectively, as

$$\mathbf{H}(Ts) = [40,400,600,90°,90°,90°]^T$$

$\mathbf{H}(Tg) = [46.283,400,600,45°,90°,90°]^T$

4.3.2 **Arm Configuration parameters**

As explained previously in section 2.1.1.1, the robot arm may be lefty or righty, i.e. it may resemble a human's left arm or right arm, respectively. The elbow can have two configurations. When the elbow's position is above the line joining the shoulder and the wrist, the elbow is up. When the position of the elbow is below that line, the elbow is down. The wrist also has two configurations. A no-flip wrist where the joint angle θ_5 is positive, and a flip wrist for which θ_5 is negative. The configuration paramenter for the arm, elbow and wrist of the robot are denoted by k_1, k_2 and k_3, respectively. Table III gives the values of these parameters used in the simulation.

Table III. Arm Configuration Parameters

Arm	Elbow	Wrist
$k_1 = -1$	$k_2 = +1$	$k_3 = -1$
righty	up	flip

4.3.3 **Stepper motor parameters**

The PUMA 560 is equipped with a DC servomotor which receives feedback in the form of pulses from a digital encoder. In this study, it is assumed that the PUMA 560 is equipped with a stepper motor whose details are given in Table IV. This is a simplifying assumption. The algorithm can be applied to feedback from digital encoders for joints that are driven by sevo motors.

Table IV. Stepper motor parameters

steps per revolution	400
speed (rps)	50
acceleration range (deg/sec/sec)	0 - 999
Maximum pulse rate (kHz)	20

4.4 **Analysis of the results**

Efficient algorithms for the approximation of cartesian space trajectories and trajectory tracking have been developed and presented here. These algorithms are based on the methods developed by [2] and [6]. A comparison between the algorithm presented by Chang et al [2] and the one in this study gave the following results.

1) In the approximation of the cartesian path segment by straight lines it was found that $| \theta^* - \theta | < \xi_2$ is not the proper criterion to be satisfied in order to obtain a line segment that approximates the arc. The original procedure is as follows:

If $| \theta^* - \theta | < \xi_2$ is not satisfied, then substitute Tf = (To+Tf)/df and repeat until the criterion is satisfied.

The problem with this step is that it will not give the optimum number of lines required to approximate the given path segment. Let us assume that $\theta^* - \theta < -\xi_2$. In this case, $| \theta^* - \theta | < \xi_2$ is not satisfied. But by substituting Tf = (To+Tf)/df we only further increase the difference between the two angles in the negative direction and this will continue until we reach the beginning point of the path itself, without obtaining a line segment. Hence, instead of coming back in time, we must go forward one step and then check again to see if the condition is satisfied. On the other hand, if $\theta^* - \theta > \xi_2$, then we must come back in time in order to reduce the difference between the two angles. One way of avoiding this problem is as follows.

If $| \theta^* - \theta | < \xi_2$ is not satisfied then

if $\theta^* - \theta < -\xi_2$ then substitute Tf = Tf + (Tf-To)/df and recalculate θ and check the above criterion again

else if $\theta^* - \theta > \xi_2$ then

substitute Tf = (Tf+To)/df and recalculate θ and check the above criterion again.

This procedure gives the correct number of line segments that can be used to approximate the cartesian path segment. Figure 7b illustrates what could happen if the original procedure is followed.

2) Chang's[2] algorithm requires eight inputs to the manipulator in the splining phase. The inputs are the velocities and accelerations at the start point, the second point, the next to the last point and the end point. Both procedures presented here require only four inputs: The velocities and accelerations at the beginning and end points, which is the normal practice.

3) By introducing a pseudo knot at the beginning of the spline, any error associated with that point will be propogated throughout the whole trajectory. That problem is avoided here by introducing both pseudo knots at the end.

4) The splining method presented here is computationally simple compared to Chang's algorithm. The splines obtained from both procedures for Joint 1 are shown in figure 8 and figure 9.

5) A comparison of figure 8 and figure 9 shows that the spline obtained as a result of procedure 2 which utilized the local smoothness constraint while calculating the coeffecients of the last two splines, is not as smooth as the spline obtained from procedure 1 which did not utilize that constraint. This would indicate that procedure 1 performs better than procedure 2. However, not enough examples were performed to draw this general conclusion. Neither one has been proven to be better than the other. In fact, both are based on similar procedures.

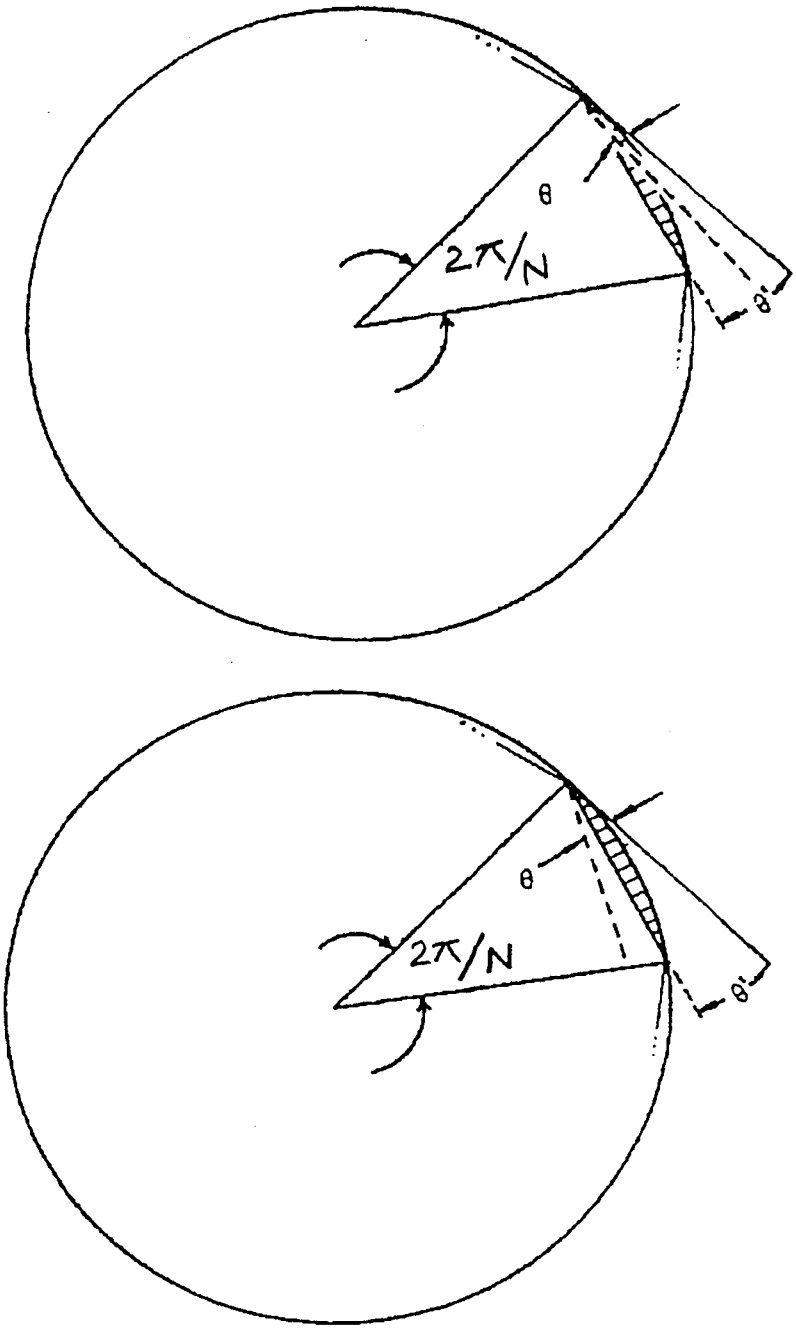


figure 7. (a) Optimum angle greater than calculated angle. (b) Optimum angle lesser than calculated angle.

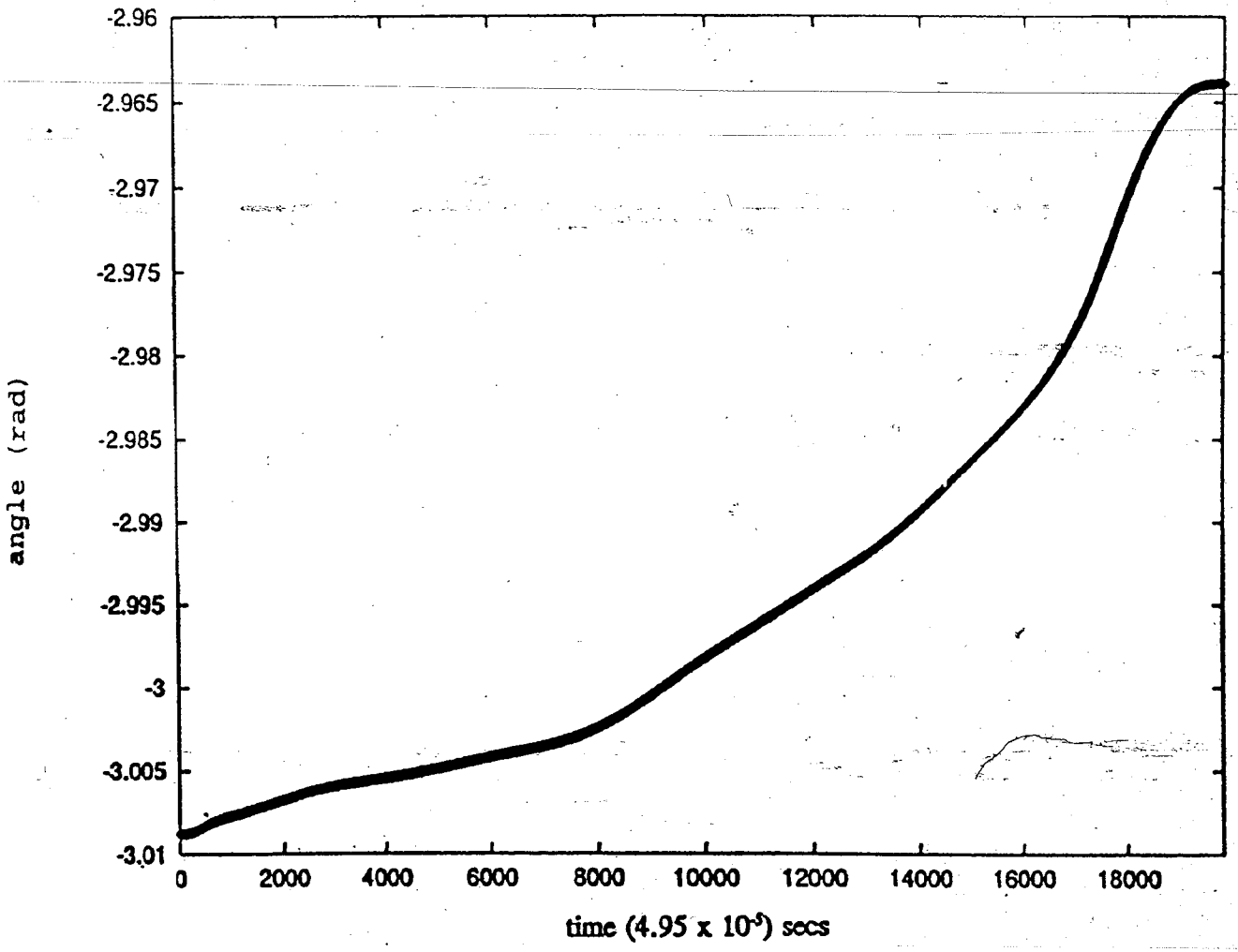


figure 8. Spline obtained from procedure 1.

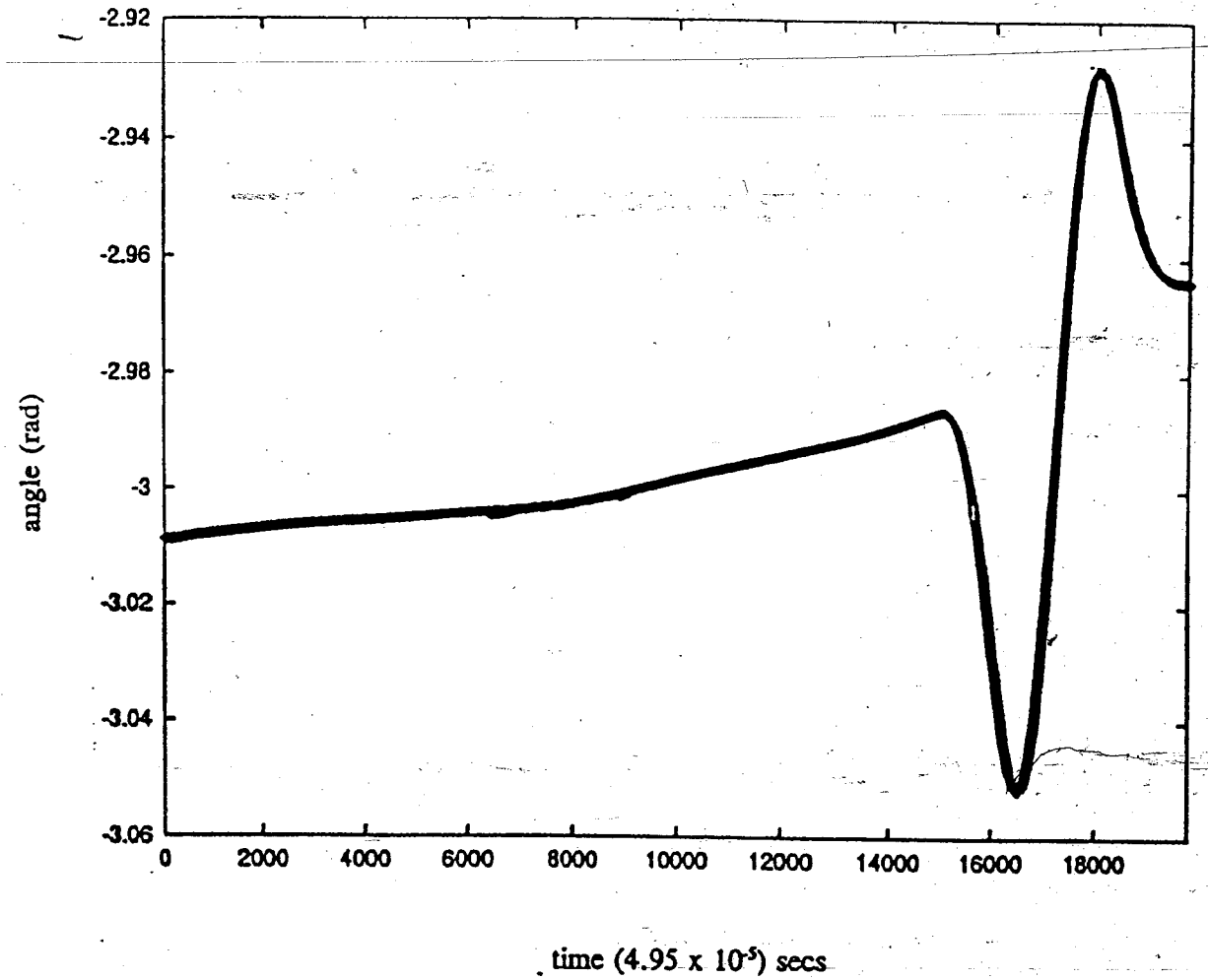


Figure 9. Spline obtained from procedure 2.

5 Conclusions

5.1 Conclusions

Efficient algorithms for the planning of cartesian trajectories and subsequent tracking in joint space have been developed. The work done in [2] and [6] forms the basis for these algorithms. A procedure for selecting knot points in cartesian space based on the curvature of the cartesian trajectory was first developed. These points were then transformed to joint space using kinematic transformations developed specially for the PUMA 560. The points in joint space were then splined using two methods. The transformation of this spline from joint space to cartesian space, results in a trajectory that closely approximates the cartesian trajectory initially specified. The algorithm in [2] has been suitably modified to give correct results. An attempt was made to simulate the algorithm in [2], but was unsuccessful. A program was written to simulate the algorithm in [2] but, the expected results were not obtained. The procedures proposed for splining of knot points in joint space are simple and do not involve complex computations. The kinematic equations used for the PUMA are also efficient and exploit the simple geometry of the arm.

The solution approach develops a good physical understanding of the manipulator and provides geometric insight into the transformation problem and also helps one to understand the difficulties involved in trajectory planning.

A comparison between the algorithms developed here and the algorithms presented in [2] gives the following results:

1) The criterion presented in [2] which is used to determine the number of line segments required to approximate the cartesian path is inadequate. The proper criterion to be used can be found in the discussion of the results.

2) The proposed splining procedures require fewer inputs than the procedure in [2].

3) The error introduced when adding a pseudo knot point at the beginning of the trajectory, which propogates throughout the splining procedure is reduced by introducing the pseudo knot point at the end.

4) The proposed splining methods are computationally simpler.

5) The procedure which did not incorporate the local smoothness criterion seems to perform better than the procedure which incorporates the smoothness criterion. But further tests must be performed before a general statement can be made.

5.2 **Directions for Future Research**

The algorithms presented here are derived assuming the manipulator to be a rigid body. Lightweight manipulator design reduces driving torques for high speed and heavy payload manipulators. However, during such operations, the manipulator is likely to deform, thereby reducing accuracy and creating stability problems. Chang and Hamilton [1], derived the kinematics of a flexible manipulator using an Equivalent Rigid Link System (ERLS) model. The ERLS is a hypothetical system whose motion and kinematics resemble that of a rigid link system.The global motion of the flexible manipulator is thus split into a large motion representing the ERLS and a superimposed small motion which is due to the deviations with respect to the ERLS.

Kinematic equations in terms of the Hartenberg-Denavit matrix have been developed and take into account the deviations due to the flexibility of the manipulator. These equations could be incorporated into the algorithm presented here in order to make it applicable to flexible manipulators.

Another consideration is the modification of the forward kinematic equations in the neighborhood of singularities. As has been reported, the accuracy of some of the other kinematic equations deteriorates in the neighborhood of a singular point. Lai and Yang[5] have proposed a scheduling method to overcome this problem.

Finally, another consideration for future research is the control of manipulator trajectory. Various schemes have been proposed, and most of them work satisfactorily. The problem is to obtain a procedure that is not computationally complex, while at the same time giving satisfactory results.

References

[1] Chang, Liang-Wey and Hamilton, J.F., "The Kinematics of Robotic Manipulators with Flexible Links using an Equivalent Rigid Link System (ERLS) Model," Journal of Dynamic Systems, Measurement, and Control, vol. 113, pp. 48-52, March 1991.

[2] Chang, Yeong-Hwa, Lee, Tsu Tian and Liu, Chang-Huan, "On-Line Approximate Cartesian Path Trajectory Planning for Robot Manipulators," IEEE Transactions on Systems, Man and Cybernetics, vol. 22, no.3, pp. 542-547, May/June 1992.

[3] Elgazaar, Shadia, "Effecient Kinematic Transformations for the PUMA 560 Robot," IEEE Journal of Robotics and Automation, vol. RA-1, no.3, pp. 142-151, September 1985.

[4] Featherstone, R., "Position and Velocity Transformations Between Robot End-Effector Coordinates and Joint Angles," International Journal of Robotics Research, vol.2, no.2, pp. 34-45, Summer 1983.

[5] Lai, Jim Z.C., and Yang, Daniel, "An Effecient Motion Control Algorithm For Robots with Wrist Singularities," IEEE Transactions on Robotics and Automation, vol.6, no.1, pp. 113-117, February 1990.

[6] Lianos, T., Kiritsis, D., and Aspragathos, N., "A Direct Algorithm for Continuous Path Control," Robotics and Computer Integrated Manufacturing, vol.8, no.2, pp. 97-101, 1991.

[7] Lin, C.-S., Chang, P.-R., and Luh, J.Y.S., "Formulization of Optimization of Cubic Polynomial Trajectories for Industrial Robots," IEEE Transactions on Automation and Control, vol. AC-28, no.12, pp. 1066-1074, December 1983.

[8] Luh, J.Y.S., and Lin, C.-S., "Approximate Joint Trajectories for Control of Industrial Robots along Cartesian Paths," IEEE Transactions on Systems, Man, and Cybernetics, vol SMC-14, pp. 444-450, May/June 1984.

[9] Paul, R.P., "Manipulator Cartesian Path Control," IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-9, pp. 702-711, November 1979.

[10] Taylor, R.H., "Planning and Executing of Straight Line Manipulator Trajectories," IBM Journal of Research and Development, vol.23, no.4, pp. 424-436, July 1979.

Vita

Name:	Dwaraka Srinivasan.
Date of Birth:	6 / 10 / 71
Current Address:	422 Pierce Street, 2nd Floor, Bethlehem, PA 18015.
Permanent Address:	9127 Edmonston Court, Apt. No. 203, Springhill Lake Apartments, Greenbelt, MD 20770.
Education:	Bachelor of Engineering, Mehanical Engineering, Coimbatore Institute of Technology, India, 1992. Master of Science, Industrial Engineering, Lehigh University, 1994.

END

OF

TITLE